CS 332 - Homework 3 - Brief Answers

Due: Monday, February 25

At the top of your homework paper write your name, the class number (CS 332) and also the homework assignment number (HW 3), and write the number of the problem at the top of your problem solution for each problem.

Reading: Chapter 4, section 4.2, pages 201-210

PROBLEMS:

1. Write out the full program of a Turing machine which decides the language \( L \) of all binary strings whose corresponding integer is one more than an even power of two.
   (That is the integers 2, 3, 5, 9, 17, ...)
   (Note: The string may have some leading 0’s before the 1.)

   You should show how the computation of the TM proceeds on an example input string, and mention the purpose/actions of the various TM states you use.

   Answer: Here is a brief description of how the TM would work.

2. Describe the computation of a TM which accepts the language
   \[ L = \{ b \mid b \text{ is a binary string which has exactly one more 1 than 0} \}. \]
   Here you can informally describe how the TM works. You need not give the full program or diagram.

3. Assume that \( S \) is a decidable set of natural numbers. Let \( S_b \) be the set of all binary strings which consists of the elements in \( S \) represented in base 2.
   i. Give an algorithm to decide \( S_b \). You may invoke Turing’s thesis here.

   ii. Is this same statement true when \( S \) is recognizable (but maybe not decidable) ?
   That is, Is it true that for any recognizable set \( S \) of natural numbers, the corresponding set \( S_b \) of binary integers is also recognizable? Explain your reasoning.
4. Show that the decidable sets are closed under intersection.
Specifically that means, show that if \( J \) and \( L \) are decided by TM’s \( M_1 \) and \( M_2 \), then there is a TM \( M_3 \) that decides \( J \cap L \).

Suggestion: For problems 4 and 5 you might have a look at the answers to these same problems for union instead of intersection in the textbook. These are problems 3.15a and 3.16a.

Answer:
Algorithm (given by a Turing machine \( M_3 \)) which decides if a string \( x \) is in \( J \cap L \).
1. Noting that both \( M_1 \) and \( M_2 \) are deciders, run \( M_1(x) \) and \( M_2(x) \) until they halt and either accept \( x \) or reject \( x \).
2. If both \( M_1(x) \) and \( M_2(x) \) both accept, then \( M_3(x) \) accepts. Otherwise \( M_3(x) \) rejects.
This \( M_3 \) is a decider which decides the language \( J \cap L \).

5. Show that the acceptable sets are closed under intersection.
Specifically show that if \( J \) and \( L \) are accepted by TM’s \( M_1 \) and \( M_2 \), there is a TM \( M_3 \) that accepts \( J \cap L \).

Answer:
Algorithm (given by a Turing machine \( M_3 \)) which accepts exactly the string in \( J \cap L \).
For \( i = 1, 2, 3, \ldots \)
1. Run both \( M_1(x) \) and \( M_2(x) \) for \( i \) steps.
2. If both \( M_1(x) \) and \( M_2(x) \) both halt and accept, then \( M_3(x) \) accepts.
3. If after \( i \) (or fewer) steps either \( M_1(x) \) or \( M_2(x) \) or both have halted and rejected then \( M_3(x) \) halts and rejects.
4. Otherwise, set \( 1 = i + 1 \) and go back to 1.
Note: that the language of \( M_3 \) is \( J \cap L \), so this set is acceptable.

6. Which two of the following 3 languages \( L_i \) are acceptable.
For each of these languages tell how a TM that accepts them would work.
If \( L_i \) is decidable given an algorithm that decides it.
\( L_1 = \{ < M > \mid M \text{ is a TM which accepts less than 3 strings} \} \)
\( L_2 = \{ < M > \mid M \text{ is a TM which accepts more than 3 strings} \} \)
\( L_3 = \{ < M > \mid M \text{ on input 0101 never moves its head past the 2nd 1 in the input} \} \)
Note: It is also true that 2 of the 3 languages are undecidable. Since we don’t yet know how to prove undecidability, you need not prove any of the languages are undecidable.

Answer: \( L_2 \) and \( L_3 \) are acceptable, \( L_1 \) is not.
Language $L_3$ is decidable (and hence also acceptable).

To accept $L_2$:
An algorithm (or TM) $T$ to accept $L_2$ would take as input a TM $<M>$ and needs to accept exactly those $M$’s which accept at least 4 input strings.

Algorithm (Turing machine) $T$ works as follows on input $<M>$.
For $i = 1, 2, 3, \ldots$
1. Run $M$ on the first $i$ strings in $\Sigma^*$ for $i$ steps each, where $\Sigma$ is the input alphabet of $M$.
(Note: What I mean by “the first $i$ inputs” here is that we can list all of the possible inputs to $M$ in length increasing order; now pick as inputs the first $i$ strings in this list. For example if $\Sigma = \{0, 1\}$ then $\Sigma^* = \{0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}$.)
2. Count how many of the $i$ inputs that $M$ runs on in step 1 cause $M$ to accept. If $M$ accepts more than 3 inputs then $T$ halts and accepts.
3. Otherwise go to step 1.
Note: that the language of $M$ is exactly those $<M>$’s which are in $L_2$, so this set is acceptable.

To decide $L_3$: (This one is tricky)
We need to test, given a TM $M$, whether or not $M$ on input 0101 never moves its tape head past the input string.

To decide this we simply run $M(0101)$ long enough to see if this happens. Long enough means enough steps to determine if $M$ is in an infinite loop during which $M$ never moves past its input.

More precisely, let $K$ be the number of 3-tuples of (different states, first 4 symbols, head position) that the TM $M$ can be in when it’s head is still on its 4 input squares. $S$ is a finite number and can be calculated.

Now run $M(0101)$ for $S+1$ steps.
If at any time up to $S+1$ steps $M$ hits a B on its tape, then it has moved past the input on the tape and we make $M$ reject.
On the other hand if $M$ halts in fewer than $S+1$ steps all within the 4 input string squares then we accept. Or if $M$ runs for $S+1$ steps and never moves past its first 4 squares then we accept because by $S+1$ steps $M$ must have repeated a full configuration and so it is in an infinite loop never leaving it’s 4 input squares.

In sum, we have an algorithms which always halts and accepts $M$ iff $M(0101)$ never leaves its first 4 tape squares during its computation.